NYU Tandon School of Engineering
CS-GY 6083, Principles of Database Systems, Fall 2024

Practice Material #4 Sample Solutions

Problem 1

1. Write the following SQL queries:

   a. Count the number of employees who work for branches in Brooklyn, NY.

   ```
   select count(distinct eID) from Payroll where branchCity = 'Brooklyn' and
   branchState = 'NY';
   ```

   b. Select the employee and correspondent working hours in March, 2024 with the highest total wage (hoursWorked * wage)

   ```
   create view employeeWage as select eID, wage*hoursWorked as monthlyWage
   from Payroll where month = 3 and year=2024;
   select distinct eID from employeeWage where monthlyWage =
   (select max(monthlyWage) from employeeWage);
   ```

   c. Output the average salary of each department.

   ```
   create view employeeWage as select eID, dept, avg(hoursWorked * wage) as
   avgWage from Payroll group by eID, dept;
   select dept, avg(avgWage) from employeeWage group by dept;
   ```

2. Anything like {eid, position} → {position} is okay
3.

   a. Branches are identified by a **branchID**, besides, the system also records associated branchCity and branchState information

   branchID → branchCity, branchState

   b. Besides, the system also records the hours worked by each employee in each month of each year.

   eID, month, year → hoursWorked

   c. No employee changes department, working branch, or position during the period covered by any payroll table.

   eID → dept, branchID, position

4. branchID → branchCity, branchState
   eID → dept, position, branchID
   eID, month, year → hoursWorked
   Position → wage
5. Yes, it is a canonical cover.
6. {eID, month, year}

7. It is not in BCNF because there are non-trivial functional dependencies in which the left-hand side is not a superkey. All of the functional dependencies listed in 4 are such “bad” functional dependencies.
8. We have:
   Branches(**branchID**,branchCity, branchState)
   Employees(**eID**, dept, position, branchID)
   WorkingHours(**eID, month, year**,hoursWorked)
   Wages(**position**,wage)

   Process:
   The candidate keys are (eid, month, year)
   Starting from branchID -> branchCity, branchState;
   We decompose Payroll into R1(branchID,branchCity, branchState) and
   R2_1(branchID,eID,dept, position,month, year,hrsWorked,wage)

   R2_1 has position → wage, which violates BCNF, so we then further decompose it into:
   R3(position,wage) and R2_2(branchID,eID,dept, position,month, year,hoursWorked)

   R2_2 has eID → dept, position, branchID, so we then decompose it into
   R4(eID, dept, position, branchID) and R5(eID,month, year,hoursWorked).

   In the end we have the following schemas:
   Branch(branchID, branchCity, branchState) [R1]
   Wages(position, wage) [R3]
   Employees(eID, dept, position, branchID) [R4]
   WorkingHours(eID,month, year,hoursWorked) [R5]

9. If we have two records in **Payroll**:
   (1,”A”,”Manager”,3,2024,1,35,154,”Brooklyn”,”NY”)
   (1,”A”,”Manager”,2,2024,1,35,151,”Brooklyn”,”NY”)
   The Projection onto the decomposed schema will be:
   Branches (1,”Brooklyn”,”NY)
   Employees (1,”A”,”Manager”,1)
   WorkingHours (1,2,2024,151) (1,3,2024,154)
   Wages (“Manager”,35)
   Join them together, the result is the same as Payroll.
10. Yes, because for each functional dependency in F+, all of the attributes of alpha and beta are contained in one of the decomposed schemas.
11.
    a. Count the number of employees who work for branches in Brooklyn, NY.
    ```
    select count(*) from Employees natural join branches where branchCity =
    'Brooklyn' and branchState = 'NY';
    ```
    b. Select the employee and correspondent working hours in March, 2024 with the highest total wage (hoursWorked * wage)

```
create view employeeWage as
select eID, wge*hrsWorked as monthlyWage from
WorkingHours natural join employees natural join Wages where month = 3
and year = 2024;
select distinct eID from employeeWage where monthlyWage =
(select max(monthlyWage) from employeeWage);
```

c. Output the average salary of each department.

```
create view employeeWage as
select eID, dept, avg(hoursWorked * wage) as avgWage
from Employees natural join WorkingHours natural join Wages group by eID,
dept;
select dept, avg(avgWage) from employeeWage group by dept;
```

12.

a. Position, branchCity, branchState → wage
b. Position → wage no longer holds.
c. No, because you would need to join Branches, Employees, and Wages to check position, branchCity, branchState → wage
d. 3NF will be:
   Branches(branchID,branchCity, branchState)
   Employees(eID, dept, position, branchID)
   WorkingHours(eID, month, year,hrsWorked)
   Wages(position, branchCity, branchState,wage)

Problem 2

A.
We have E->A, G; F->B, C, G; plus D, we have {D, E, F} -> {A, B, C, D, E, F, G}
All candidate keys: {D, E, F}

B.
Because of F → B, B is extraneous in FB → C.
Because of D → A, F is extraneous in DF → A.
So the canonical cover will be:
Fc = { D→A, E→AG, F→BCG}

c.
It is not in BCNF because there is nontrivial functional dependency: F → C and F is not the superkey. Decomposing relation R into BCNF we get:
Step 1 : R1 = {A, D} and R2 = {B, C, D, E, F, G}
Step 2 : R2 can be further decomposed into R2 and R3 because F→BCG is nontrivial functional dependency and D is not a candidate key.
Hence BCNF form is:
R1 = {A, D}

R2 = {B, C, F, G}
R3 = {D, E, F}

d.
No. The BCNF form in c is not dependency preserving because we cannot check E → AG in the result of (c) without joining two of the decomposed tables.
Converting into 3NF we get:
Fc = { D→A, E→AG, F→BCG} and Candidate key = {D, E, F}
Therefore, 3NF form is:
R1 = {A, D}
R2 = {A, E, G}
R3 = {B, C, F, G} and
R4 = {D, E, F}